Final

- Please read all instructions (including these) carefully.
- There are seven questions on the exam, some with multiple parts. *Question 3d is for extra credits.* You have 2:00 to work on the exam.
- The exam is open book, open notes.
- Please write your answers in the space provided on the exam, and clearly mark your solutions. You may use the backs of the exam pages as scratch paper. Please do not use any additional scratch paper.
- Solutions will be graded on correctness and clarity. Each problem has a relatively simple and straightforward solution. You may get as few as 0 points for a question if your solution is far more complicated than necessary. Partial solutions will be graded for partial credit.

NAME: _____
First Name Last Name

Problem	Max. points	Points
1	15	
2	12	
3	20 + 10	
4	10	
5	24	
6	10	
7	9	
TOTAL	100 + 10	

1. Inheritance in object-oriented languages (15 points)

   We have four classes: A, B, C, and D. The following are contents of their virtual method tables (VMTs):

```
A's VMT                    B's VMT
-------                    -------
  A::m1                      B::m1
  B::m2                      B::m2
  C::m3
  A::m4

C's VMT                    D's VMT
-------                    -------
  C::m1                      B::m1
  B::m2                      D::m2
  C::m3                      D::m3
```

   (a) (5 points) Draw the inheritance graph to relate the four classes A, B, C, and D (put the base class at the top).

(b) (8 points) Complete the code for the four classes. Assume that the classes do not have any instance variables and all methods are of the form:

```
void foo() {} /* empty body */
```

Specify class inheritance using the `extends` keyword where necessary.

```
class A                    {




}

class B                    {



}
class C                    {

}
class D                    {



}
```


(c) (2 points) In this part, suppose we make some modifications to the class definitions by adding a static method `m5` to class A. Would this change affect A's VMT?
(circle one): **YES** or **NO**.
If you circled **YES**, show the new VMT for A.
If you circled **NO**, explain why that has no effect.

2. Semantics of LISP programs (12 points)

Evaluate the following s-expressions and show the output of the last s-expression. Circle your answers.

(a) (1 point)

```
(* (- (+ 2 5) 4) 8)
```

(b) (1 point)

```
(cdr (car (cdr '(1 (2 3) 4))))
```

(c) (3 points)

```
(do ((l1 '(8 7 6 5 4 3 2 1) (cdr l1))
     (l2 '(1 2 3 4 5 6 7 8) (cdr l2))
     (cnt 0 (+ 1 cnt)))
    ((< (car l1) (car l2))
     cnt))
```


(d) (3 points)

```
(let ((a 2)
      (b 4))
     (defun foo (c)
        (let ((b 6))
          (list a b c))))
(let ((a 8)
      (b 10))
     (foo b))
```

(e) (4 points)

```
(defun foo (f l)
  (if (null l) ()
    (let ((l2 (mapcar f l)))
      (cons (car l2) (foo f (cdr l2))))))
(foo (lambda (x) (+ x x)) '(1 2 4))
```

3. LISP function definitions (20 + 10 points)

   In this problem, you are asked to write a few LISP functions to perform certain tasks.

   (a) (6 points) Write a function, `mymapcar`, to implement the built-in `mapcar` function.

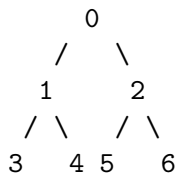
   (b) (7 points) Write a function, `deleteall`, that takes two arguments, a list `lst` and an atom `ele`, and returns the original list with all occurrences of `ele` deleted, including all nested ones.


```
> (deleteall '(a 4 b 4 c) 4)
(A B C)
> (deleteall '(a (1 a b) ((a))) 'a)
((1 B) (NIL))
```

(c) (7 points) Write a function, `sublist`, that takes as arguments two lists of numbers or symbols `l1` and `l2` and returns true if `l1` is a subsequence of `l2`. It has similar functionality as the Prolog built-in predicate `sublist`.

```
> (sublist '(a b) '(c a 2 b d))
T
> (sublist '(a b c) '(a c b))
NIL
```

(d) **Extra Credits** (10 points) In LISP, we can represent a binary trees with an s-expression where the first and third elements are the left and right subtrees, and the second element is the label of the node. For example, the empty tree is represented as `()`, the tree



is represented as `((() 3 ()) 1 (() 4 ())) 0 ((() 5 ()) 2 (() 6 ()))`.

Write a function, `levelorder`, that traverses a well-formed binary tree expression (as above) in level order and prints a list of the visited nodes in order. So for the example tree the output would be `(0 1 2 3 4 5 6)`.

4. Prolog search and backtracking (10 points)

The following describes a set of rules and facts:

rule 1	`a(X,Y) :- b(Y), c(X), d(X,Y).`
rule 2	`b(0).`
rule 3	`b(1).`
rule 4	`c(2).`
rule 5	`d(0,1).`
rule 6	`d(2,1).`

On the lines below, give a step by step description of how Prolog evaluates the following query:

`| ?- a(X,Y).`

Your answer must be descriptive. At each step show the following: (i) the subgoals at the step; (ii) the rule that will be applied (write "none" if no of the rules applies); (iii) variables instantiations, if any; and (iv) comments that indicate if there was a "match", "success", "failure", or "backtrack". Clearly mark the line where "success" is achieved and show the output generated.

Also, when backtracking, clearly show the goals to which the system backtracks. Continue on the back of the page if you need more space.

step no.	current subgoals	rule applied	variable instantiation	comments
1	`a(X,Y)`	1		match
2	`b(Y), c(X), d(X,Y)`	2	`Y=0`	match

5. Prolog programs (24 points)

   (a) (6 points) What do the following Prolog queries output given their associated facts and rules?

      i. (2 points)

```
| ?- (X = 0 ; X = 1), !, X = 1.
```

      ii. (2 points)

```
foo(X,[X|_]) :- !.
foo(X,[_|T]) :- foo(X,T).
bar([],0).
bar([H|T],R) :- bar(T,R1), R is R1+1.
| ?- foo(0,[1,0]), !, bar(X,2).
```


      iii. (2 points)

```
guesswhat([X, _ | Y], [X | Z]) :- !, guesswhat(Y, Z).
guesswhat(_, []).
| ?- guesswhat([1,2,3,4,5], R).
```

(b) (6 points) Write a predicate, `perm(L, S)`, which is true if given a list `L`, `S` is a permutation of `L`.

```
| ?- perm([], S).
S = []
yes
| ?- perm([1,2,3], S).
S = [1,2,3] ? ;
S = [2,1,3] ? ;
S = [2,3,1] ? ;
S = [1,3,2] ? ;
S = [3,1,2] ? ;
S = [3,2,1] ? ;
no
```


(c) (6 points) Write a predicate, `gcd(X, Y, R)`, which is true if given two integers, `X` and `Y`, `R` is their greatest common divisor (GCD).

```
| ?- gcd(12, 18, R).
R = 6
yes
| ?- gcd(3, 5, R).
R = 1
yes
```

(d) (6 points) Write predicates to implement the common Boolean operators `and(X,Y)`, `or(X,Y)`, `negate(X)`, and `imply(X,Y)`.

6. Unification in Prolog (10 Points)

For each Prolog expression, fill the variable instantiations (only for those variables used in the expression) or put a check in the "no" column if evaluation fails. The first line is already filled correctly as an example. The expressions are independent of each other. *(No partial credit, no work needs to be shown.)*

query	X =	Y =	Z =	no
`X = 3, Y is X*X.`	3	9		
`X = a, Y = b, Z = g(X, Y, Y).`				
`X = 0, X-Z = (Y+2)-Z, Z=X+1.`				
`[[a|X], X, Y] = [Z, [a,b], [Z]].`				
`g(c, f(e,Y), [d|X]) = g(Y, f(X,c), [d, e]).`				
`[f(a,X)|Z]=[Y|g(w,g)],[X|f(a,Y)]=[f(a,b,c)|f(a,Y)].`				

7. Parameter passing (9 points)

Consider the following program in C-like syntax:

```
#include <stdio.h>

void foo (int x, int y)
{
  int t = y;
  y = x;
  x = t;
}

void main ()
{
  int A[3] = {1, 2, 3}, k = 0;

  foo (A[k], k);

  printf("%d, %d, %d, %d", k, A[0], A[1], A[2]);
}
```


What does the program output under the following different parameter passing schemes?

(a) (3 points) call-by-value:

(b) (3 points) call-by-reference:

(c) (3 points) call-by-name: